

DAA C2 Assignment-1

Group Members –

IIT2019108 - Harsha Vardhan Madasi

IIT2019109 - Rahul kumar

IIT2019110 - Sumit Katiyar

21 March 2021

Question -

Given an array of $2n$ elements in the following format $a_1, a_2, a_3, a_4, \dots, a_n, b_1, b_2, b_3, b_4, \dots, b_n$.

The task is shuffle the array to $a_1, b_1, a_2, b_2, a_3, b_3, \dots, a_n, b_n$ without using extra space.

Example -

INPUT :

8

1 3 5 7 2 4 6 8

OUTPUT:

1st iteration: 1 3 2 4 5 7 6 8

2nd iteration: 1 2 3 4 5 7 6 8

3rd iteration: 1 2 3 4 5 6 7 8

Final Shuffled Array : 1 2 3 4 5 6 7 8

Some Definitions:-

Divide and Conquer Algorithm:- A divide-and-conquer algorithm recursively breaks down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

Algorithm Description:-

Here we are using Divide and Conquer Technique.

Firstly we will divide the given array into half and swap second half element of arr1[] with first half element of arr2[]. Recursively do this for both the arrays arr1 and arr2.

Algorithmic Steps:-

- Let us consider an array be {a1, a2, a3, a4, b1, b2, b3, b4} .
- Now split the array into two halves i.e., {a1, a2, a3, a4} ; {b1, b2, b3, b4}.
- Now we will swap elements which are present around the center i.e., we will swap a3, a4 with b1, b2 correspondingly.
- After swapping we will get: a1, a2, b1, b2, a3, a4, b3, b4.
- Now we will recursively split the above subarrays and swap the elements around the center for each subarray.
- Finally we will get a1, b1, a2, b2 and a3, b3, a4, b4.

Pseudo Code:-

```
//Function to shuffle an array  
void shufflearray(int arr[],int start , int end)  
// base condition  
if(end < start )then  
return ;  
//if only two elements present in the subarray  
if(end = start + 1)then  
return ;
```

Psuedo Code(contd.):

//Finding middle of the array to divide the array

int mid \leftarrow (start + end) / 2;

//using temp in order to swap first half of second array

int temp \leftarrow mid + 1;

//using first in order to swap second half of first array

int firstmid \leftarrow (start + mid) / 2;

//Swapping the center elements

for(i = firstmid + 1 to i \leq mid)

Pseudo Code(contd.):-

```
swap (arr[i],arr[temp++]);
```

```
i ← i+1;
```

```
// recursively calling the function for first and second subarrays
```

```
shufflearray(arr, start, mid);
```

```
shufflearray(arr, mid + 1, end);
```

```
//MAIN FUNCTION
```

```
int main()
```

```
int n;
```

```
input n;
```

```
int arr[n];
```

Pseudo Code(contd.):-

```
//taking input  
for(int i = 0 to n )  
input arr[i];  
shuflearray (arr,0,n-1);  
//printing the final shuffled array  
for(int i = 0 to n )  
print arr[i];  
return 0;
```

Time Complexity:-

In the above algorithm we get the recurrence

$$T(n) = 2T(n/2) + O(n)$$

for the time complexity, which results in

$$T(n) = O(n \log n)$$

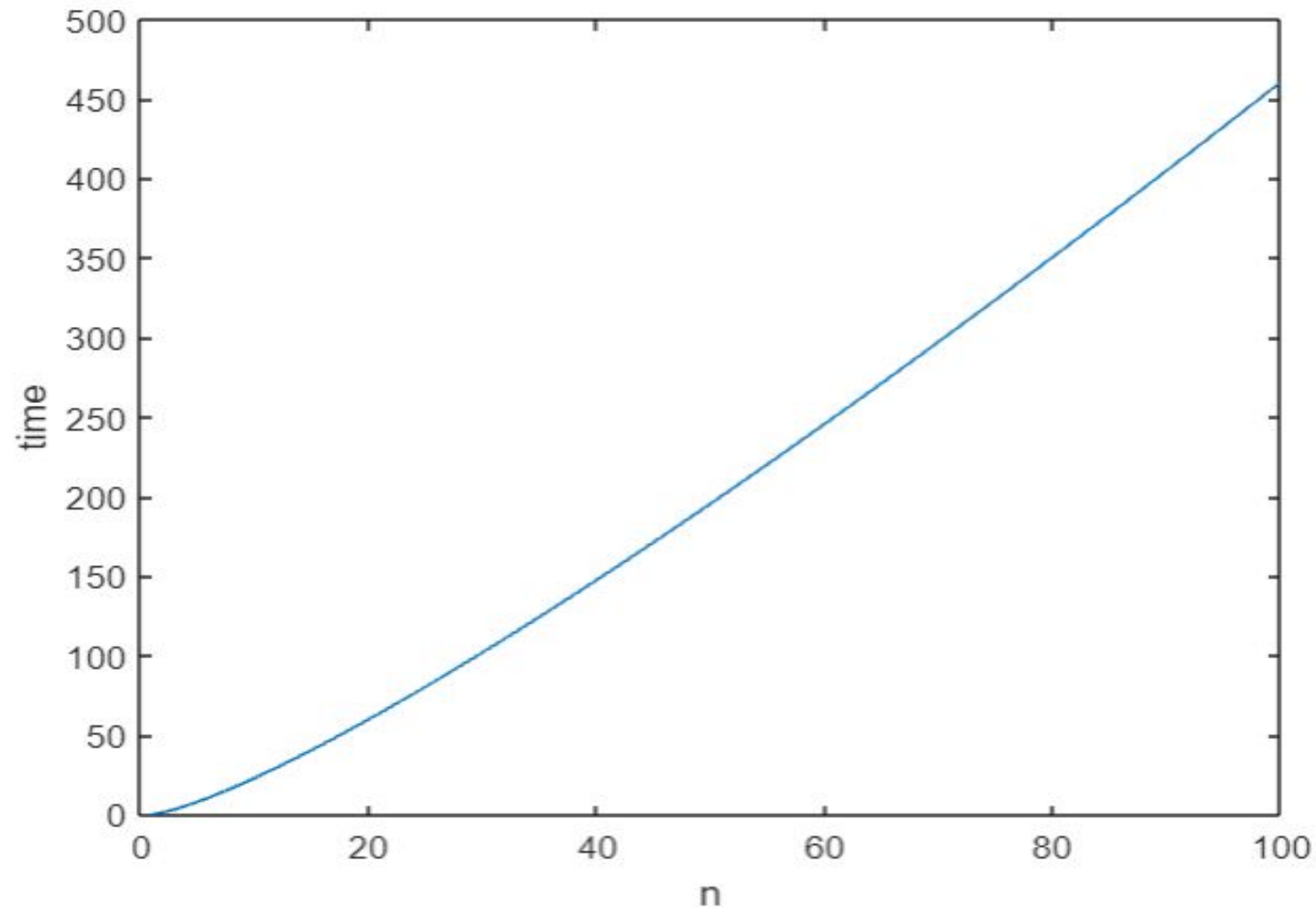
So the final computational time to shuffle the given array is $O(n \log n)$.

Space Complexity:-

To be precise , as we are swapping the elements in the given array itself there is no extra space required.

So the space complexity will be $O(1)$.

Time Complexity Graph($O(n \log n)$):-



Conclusion:-

Hence, from the above graph and analysis, we come to the conclusion that this problem can be solved using the time complexity of $O(n \log n)$ and space complexity of $O(1)$.

References:-

- 1) Introduction to Algorithms by Cormen, Charles, Rivest and Stein.
- 2) Geeks for Geeks.

Thank You